REMARKS

This Amendment responds to the Office Action dated October 22, 2003 in which the Examiner rejected claims 1-24 under 35 U.S.C. §103.

As indicated above, claims 1 and 11 have been amended to make explicit what is implicit in the claims. The remaining claims have been amended to correspond to claims 1 and 11. It is respectfully submitted that the amendment is unrelated to a statutory requirement for patentability and does not narrow the literal scope of the claims.

Claims 1 and 11 claim a data processing system comprising a plurality of processors and a memory. The plurality of processors execute a series of different types of processing functions on data to be processed in a prescribed order. Each processor executes a processing function different from one another. The data to be processed is image data that consists of a plurality of pixel data. The memory stores the data to be processed in association with state information to represent the processing to be performed next for each pixel data to be processed. Processing functions are asynchronously executed on the data to be processed by the plurality of processors. One processing is executed on each pixel data by one processor at a time. The plurality of processors share the memory.

Through the structure of the claimed invention a) having each processor execute a processing function different from one another and b) having one processing executed on each pixel data by one of the processors at a time, as claimed in claims 1 and 11, the claimed invention provides a data processing system capable of processing data at high speed while allowing the memory capacity to be reduced. The prior art does not show, teach or suggest the invention as claimed in claims 1 and 11.

Claims 1-24 were rejected under 35 U.S.C. §103 as being unpatentable over *Orimo et al.* (U.S. Patent No. 5,630,135) in view of *Valentaten et al.* (U.S. Patent No. 5,250,940).

Orimo et al. appears to disclose the term of "multiple-version programs" means a plurality of programs for performing the same function but having different program structures. The execution results of the programs may be either the same or not the same. (col. 1, lines 18-22) In a distributed processing system having a plurality of processors connected through a network, at least two first processors execute multiple-version programs which perform the same function, and messages which contain data output as execution results of the programs and attribute information indicating the versions of the executed programs are sent from the first processors to the network. The messages containing the results of processing by the multiple-version programs sent from the first processors are received by a second processor, which selects one message from the received messages based on the attribute information contained in the received messages and executes a program in the second processor by using the data contained in the selected message. (col. 2, lines 2-16) As shown in Fig. 1, multiple-execution system 100 for the multiple-version programs comprises processors 11, 12, 13 and 14 connected to any type of network. Each of the processors 11, 12, 13 and 14 stores an application program in an associated internal memory and executes the application program to conduct various processings. Each application program has a name assigned to correspond to a function and attribute, including version information and is managed thereby. Where the multiple-version programs performing the same function are not present, the attribute thereof is

"Null". (col. 3, lines 57-67) In FIG. 10, application programs 71a, 71b and 71c of different versions are arranged in the processors 11, 12 and 13, respectively. The application programs 71a, 71b and 71c are those of different versions which perform the same function, and the names thereof are "X" and the attributes are "1", "2" and "3", respectively. The application program 71a performs a simulation at a high precision based on a complex logic and has a long calculation time. On the other hand, the application program 71c performs a simulation at a low precision based on a simple logic and a calculation time thereof is short. The precision and the calculation time of the application program 71b are in between those of the application program 71a and the application program 71c. Application programs 72 and 73 are arranged on the processor 14 and a terminal 1401 is connected to the processor 14 as interface means with a user. The application program 72 requests the simulation based on a request inputted from the terminal 1401. The application program 73 receives the result of the simulation to output it to the terminal 1401. When the processor 14 receives the request from the terminal 1401, it executes the processing by the application program 72 and outputs a message 720 requesting the simulation to the network 1. When the processors 11, 12 and 13 receives the message 720, they start the execution of the application programs 71a, 71b and 71c, respectively. The execution result of the application program 71c is first outputted to the network 1 as a message 710c. Thereafter, the execution result of the application program 71b is outputted to the network 1 as a message 710b, and the execution result of the application program 71a is finally outputted to the network as a message 710a. The processor 14 receives the messages 710c, 710b and 710a in this sequence and executes the

processing by the application program 73 each time, it receives the application program and displays the content of the received message on the terminal 1401. In this manner, the result of the low precision simulation is first displayed on the terminal 1401, and then the result of the middle precision simulation is displayed. Finally, the result of the high precision simulation is displayed. Thus, the user can previously acquire a general result before he/she acquires the result of the high precision simulation. Since the user can grasp the relation between the results of gradually increasing precision, the user interface is improved. For example, assuming that the application programs 71a, 71b and 71c are image analysis programs of different analysis precisions, a coarse image is displayed on the terminal 1401 shortly after the request for process, and the image of higher precision are displayed as the time elapses. (Col. 9, line 55 through Col. 10, line 36) In accordance with the program execution method and the processing system, the programs which perform the same function but have different execution logics or execution precisions are executed in parallel, and one of the execution results is selected based on the versions of the programs, the contents of the execution results and the output times. (Col. 10, lines 50-56)

Thus, *Orimo et al.* merely discloses that at least two of the processors 12 and 13 execute different versions of the same program which perform the same function (see Col. 1, lines 18-20, Col. 9, lines 57-59 and Col. 10, lines 51-56). Thus, nothing in *Orimo et al.* shows, teaches or suggests that each processor executes a processing function different from one another as claimed in claims 1 and 11. Rather, *Orimo et al.* merely discloses executing different versions of the same program (i.e. same processing functions).

Additionally, *Orimo et al.* merely discloses that the processors 12 and 13 process the same data simultaneously (col. 8, lines 34-37, col. 10, lines 11-13). However, as claimed in claims 1 and 11, one processing is executed on each pixel data by one processor at a time. However, *Orimo et al.* teaches away from the claimed invention and processes the data simultaneously in processors 12 and 13.

Valentaten et al. appears to disclose a processing system for home terminals that utilizes a single embedded processor and a single random access memory (RAM) for performing not only video/graphics tasks for a wide range of video/graphics standards and modes, but also all functions and controls for a variety of modem, voice algorithm and general purposes operations. (Col. 1, lines 10-16) Referring to FIG. 5, both the DSP module 26 and the general purpose processor 28 are connected to an internal bus 30, allowing both the DSP module 26 and the general purpose processor 28 to communicate with the RAM 12 and ROM 20 via a conventional bus interface unit 32 for transfer of control/status information and addresses/data therebetween. It will be understood by those skilled in the art that the internal bus 30 comprises both an internal address bus for handling address references by the DSP module 26 and the general purpose processor 28 and an internal data bus for handling instruction and data transfers. To save bus bandwidth, the DSP module 26 stores operands used in executing DSP algorithms in an internal RAM memory array 34 which is also accessible to general purpose processor 28. That is, the internal memory array 34 serves as a shared resource for both the DSP module 26 and the general purpose processor 28. (Col. 5, lines 45-62) The DSP module 26 may fetch operands in parallel from the internal memory array 34 and the system memory, i.e.

RAM 12. The DSP module 26 executes vector operations on complex variables that are optimized for DSP applications. The general purpose processor 28 treats the DSP module 26 as a memory mapped I/O device that occupies a reserved memory space, interfacing with the DSP module 26 via a set of memory mapped registers. (Col. 6, lines 3-11) In the operation of the embedded processor 14, the general purpose processor 28 selects from a basic set of DSP operations to define a specific sequence of operations as the DSP algorithm to be executed by the DSP module 26 for recovering data from the incoming digital signal. The general purpose processor 28 then accesses RAM 12 to retrieve operands required for execution of the selected DSP algorithm and/or instructions and data critical to the general purpose processor 28 for controlling the DSP module 26 or for performing general purpose tasks and loads them into the internal RAM array 34. The general purpose processor 28 then invokes the first DSP operation in the selected sequence by issuing the corresponding command to the control register of the DSP module 26. The DSP module 26 then places the general purpose processor 28 in a continuous WAIT state while it performs the first DSP operation utilizing operands retrieved by the address generator 38 from the internal array 34 and system memory 12. Upon completion of the DSP operation, the DSP module 26 cancels the continuous WAIT state and the general purpose processor 28 then either reads the status of the DSP module 26 or the result of the DSP operation or carriers on with the execution of its normal program flow, which may be either invoking the next DSP operation in the selected sequence by issuing the appropriate command to the DSP module control register or performance of a general purpose task. This process continues until the selected sequence of DSP operations has been completed.

The general purpose processor may then download the contents of the shared internal RAM array 34 and retrieve a new set of operands, instructions and data for further DSP operations or general purpose processing tasks. (Col. 6, lines 21-54)

Thus, *Valentaten et al.* merely discloses a digital signal processor (DSP) module 26 stores operands in an internal RAM memory array 34 which is also accessible to a general purpose processor 28. In addition, the general purpose processor 28 controls the DSP module 26 (column 6, lines 26-35). Thus, nothing in *Valentaten et al.* shows, teaches or suggests a) a plurality of processors which execute a series of different types of processing functions or b) processing is executed on each pixel data by one of the processors at a time as claimed in claims 1 and 11. Rather, *Valentaten et al.* merely discloses a memory array 34 which is accessible to both a general purpose processor 28 and the DSP module 26 which the processor 28 controls.

The combination of *Orimo et al.* and *Valentaten et al.* would merely suggest that the processors which perform the same function of *Orimo et al.* control a DSP processor 26 with which they share a memory as taught by *Valentaten et al.* Thus, nothing in the combination of *Orimo et al.* and *Valentaten et al.* shows, teaches or suggests each processor executes a processing function different from one another as claimed in claims 1 and 11. Therefore, applicants respectfully request the Examiner withdraws the rejection to claims 1 and 11 under 35 U.S.C. §103.

Claims 2-10 and 12-24 depend from claims 1 and 11 and recite additional features. Applicants respectfully submit that claims 2-10 and 12-24 would not have been obvious within the meaning of 35 U.S.C. §103 over *Orimo et al.* and *Valentaten et al.* at least for

the reasons as set forth above. Therefore, applicants respectfully request the Examiner withdraws the rejection to claims 2-10 and 12-24 under 35 U.S.C. §103.

Thus it now appears that the application is in condition for reconsideration and allowance. Reconsideration and allowance at an early date are respectfully requested. Should the Examiner find that the application is not now in condition for allowance, applicants respectfully request the Examiner enters this Amendment for purposes of appeal.

If for any reason the Examiner feels that the application is not now in condition for allowance, it is respectfully requested that the Examiner contact, by telephone, the applicants' undersigned attorney at the indicated telephone number to arrange for an interview to expedite the disposition of this case.

In the event that this paper is not timely filed within the currently set shortened statutory period, applicants respectfully petition for an appropriate extension of time. The fees for such extension of time may be charged to our Deposit Account No. 02-4800.

In the event that any additional fees are due with this paper, please charge our Deposit Account No. 02-4800.

Respectfully submitted,

BURNS, DOANE, SWECKER & MATHIS, L.L.P.

By: _____
Ellen Marcie Emas
Registration No. 32,131

P.O. Box 1404
Alexandria, Virginia 22313-1404
(703) 836-6620
Date: January 14, 2004